Newton School of Technology

15

Lists in Python-3

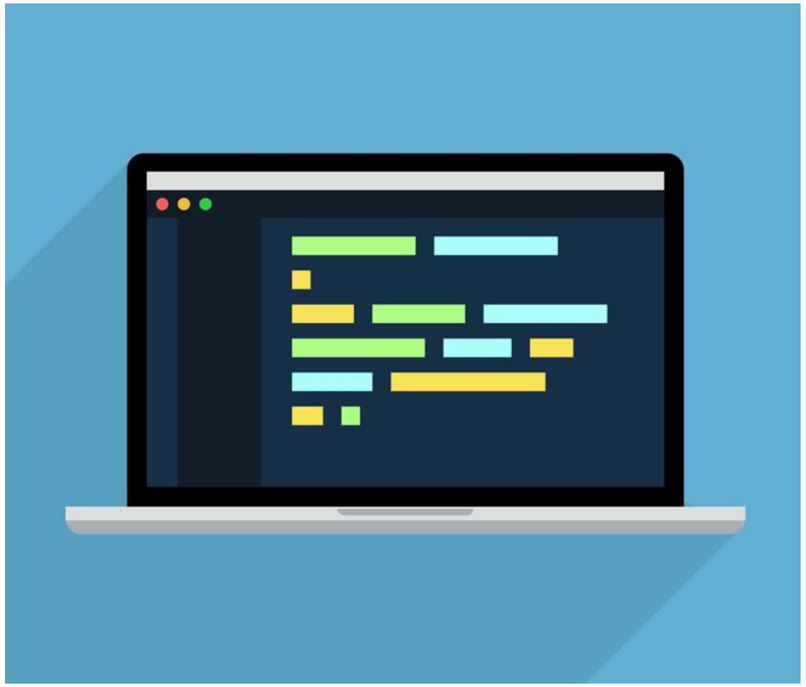
Newton School of Technology

Recap of Previous Lecture!

Quick Recap :

Newton School of Technology

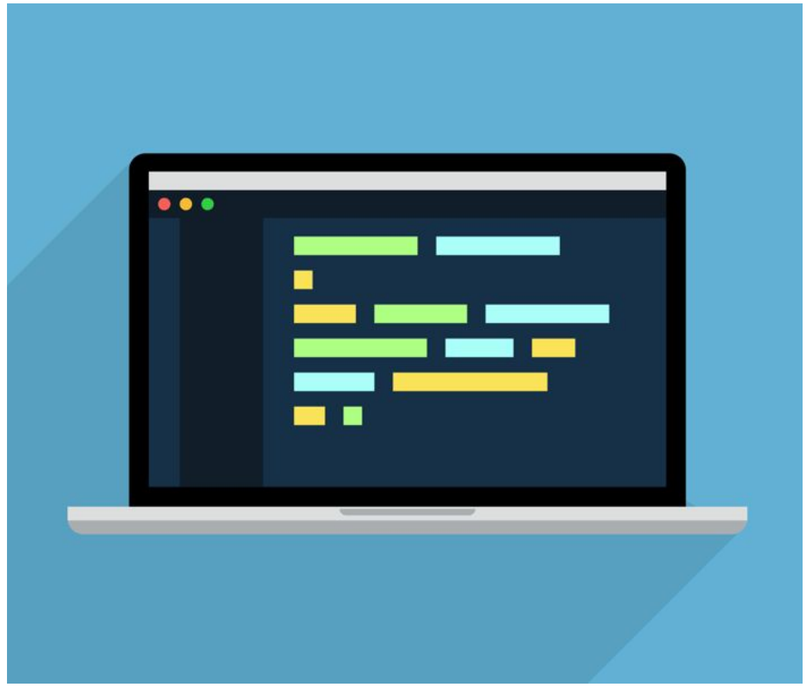
- Iterating Lists
  - Iterating With Index
    - With for loop
    - With while loop
  - Iterating Without Index



Quick Recap - Continued:

Newton School of Technology

- **Built-In functions with list**
  - **max(), min()**
  - **sum()**
  - **any(), all()**



Newton School of Technology

Important List methods

sort() function :

Newton School of Technology

This **modifies** the **original** list to sort its elements in **ascending** order.

```
my_list = [3, 1, 4, 1, 5, 9, 2]
my_list.sort()
print(my_list)  # Output: [1, 1, 2, 3, 4, 5, 9]
```

sorted() function :

Newton School of Technology

This function **returns** a **new list** with the elements sorted **without modifying** the original list.

```
my_list = [3, 1, 4, 1, 5, 9, 2]
sorted_list = sorted(my_list)
print(sorted_list)  # Output: [1, 1, 2, 3, 4, 5, 9]
```

reverse() function :

Newton School of Technology

This method **reverses** the elements of a list **in place**.


```
my_list = [1, 2, 3, 4, 5]
my_list.reverse()
print(my_list)  # Output: [5, 4, 3, 2, 1]
```

count() function :

Newton School of Technology

This method **counts** the number of **occurrences** of a specified **element** in a list.

```
my_list = [1, 2, 2, 3, 2, 4, 2, 5]
count_of_twos = my_list.count(2)
print(count_of_twos)  # Output: 4
```

index() function :

Newton School of Technology

This method **returns** the **index** of the **first occurrence** of a specified **element** in a list.

```
my_list = ['a', 'b', 'c', 'b', 'd']
index_of_b = my_list.index('b')
print(index_of_b)  # Output: 1
```

Newton School of Technology

Question 1 – Purav The Data Analyst

Newton School of Technology

List Comprehensions

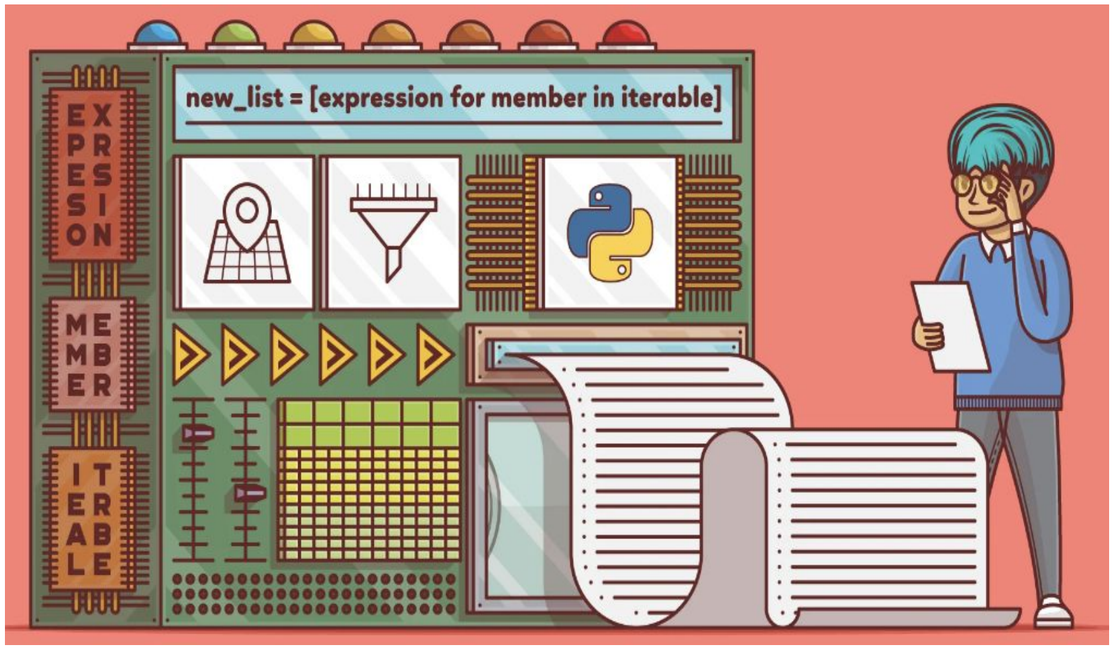
List Comprehensions :

Newton School of Technology

List comprehensions provide a concise way to create lists in Python.

They allow you to **create a new list** by **applying an expression** to each item of an iterable.

Enables filtering the items **based on a condition**.

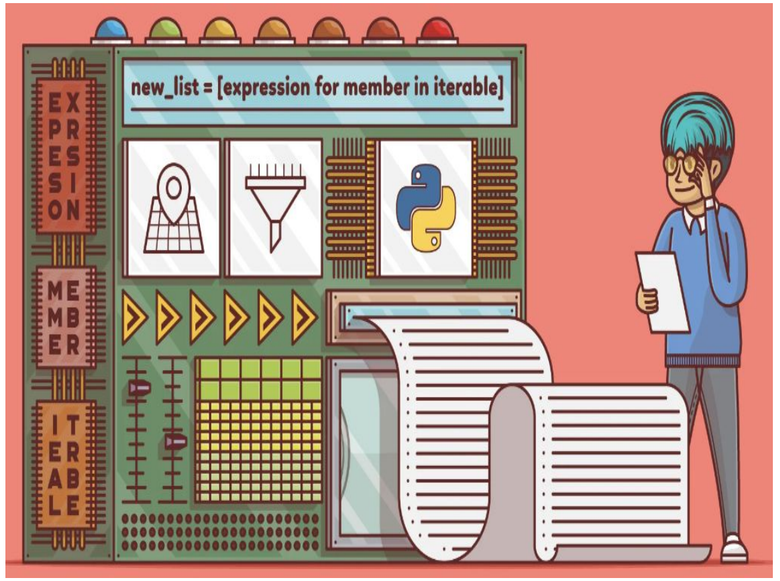


List Comprehensions :

Newton School of Technology

Syntax

```
[expression for item in iterable if condition]
```



`expression`: This is the expression to **evaluate and then include** in the new list.

`iterable`: This is the **sequence** of items that you want to iterate over (e.g., a list, tuple, range, etc.).

List Comprehensions -1 :

Newton School of Technology

```python
# Using a for loop
squares = []
for x in range(1, 6):
    squares.append(x ** 2)
print(squares)  # Output: [1, 4, 9, 16, 25]
```

List Comprehensions -1 :

Newton School of Technology

```
# Using a list comprehension
squares = [x ** 2 for x in range(1, 6)]
print(squares)  # Output: [1, 4, 9, 16, 25]
```

List Comprehensions -2 :

Newton School of Technology

```
# Using a for loop
evens = []
for x in range(1, 11):
    if x % 2 == 0:
        evens.append(x)
print(evens)  # Output: [2, 4, 6, 8, 10]
```

List Comprehensions -2 :

Newton School of Technology

```
# Using a list comprehension
evens = [x for x in range(1, 11) if x % 2 == 0]
print(evens)  # Output: [2, 4, 6, 8, 10]
```

Newton School of Technology

Let’s Practice on the Playground!

List Comprehensions -3 :

Newton School of Technology

```
words = ["hello", "world", "how", "are", "you"]
```

Make a new list which has words containing more than 3 letters.

```
long_words = [word for word in words if len(word) > 3]
print(long_words)  # Output: ['hello', 'world']
```

List Comprehensions -4 :

Newton School of Technology

```
words = ["hello", "world", "how", "are", "you"]
```

Make a new list which has every word in uppercase.

```
# Example 1: Convert each word to uppercase
uppercase_words = [word.upper() for word in words]
print(uppercase_words)  # Output: ['HELLO', 'WORLD', 'HOW', 'ARE', 'YOU']
```

Newton School of Technology

Question 2 – Square of Evens

Newton School of Technology

Nested Lists

Nested Lists

Newton School of Technology

Nested lists are lists that contain one or more other **lists as their elements**.

```
nested_list = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]


# Accessing elements
print(nested_list[0])
print(nested_list[1][1])
print(nested_list[2][0])
```

Nested Lists

Newton School of Technology

Nested lists are lists that contain one or more other **lists as their elements**.

```
nested_list = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]


# Accessing elements
print(nested_list[0])       # Output: [1, 2, 3]
print(nested_list[1][1])    # Output: 5 (accessing element at row 1, column 1)
print(nested_list[2][0])    # Output: 7 (accessing element at row 2, column 0)
```

Use Cases : Matrices

Newton School of Technology

Nested lists are commonly used to represent **matrices** in mathematics or grids of data.

```
matrix = [[1, 2, 3],
          [4, 5, 6],
          [7, 8, 9]]
```

Use Cases : Hierarchical Data

Newton School of Technology

They are useful for representing **hierarchical data** structures where each level of the hierarchy can contain **multiple elements** or substructures.

```
person1 = ['John', 25, 'Engineer']
person2 = ['Jane', 30, 'Doctor']
people = [person1, person2]
```

Use Cases : Table Like Data

Newton School of Technology

When working with **tabular data** that has **rows and columns**, nested lists can be a natural choice.

```
table_data = [
    ['Name', 'Age', 'Occupation'],
    ['John', 25, 'Engineer'],
    ['Jane', 30, 'Doctor']
]
```

Newton School of Technology

Question 3 – Book Catalog

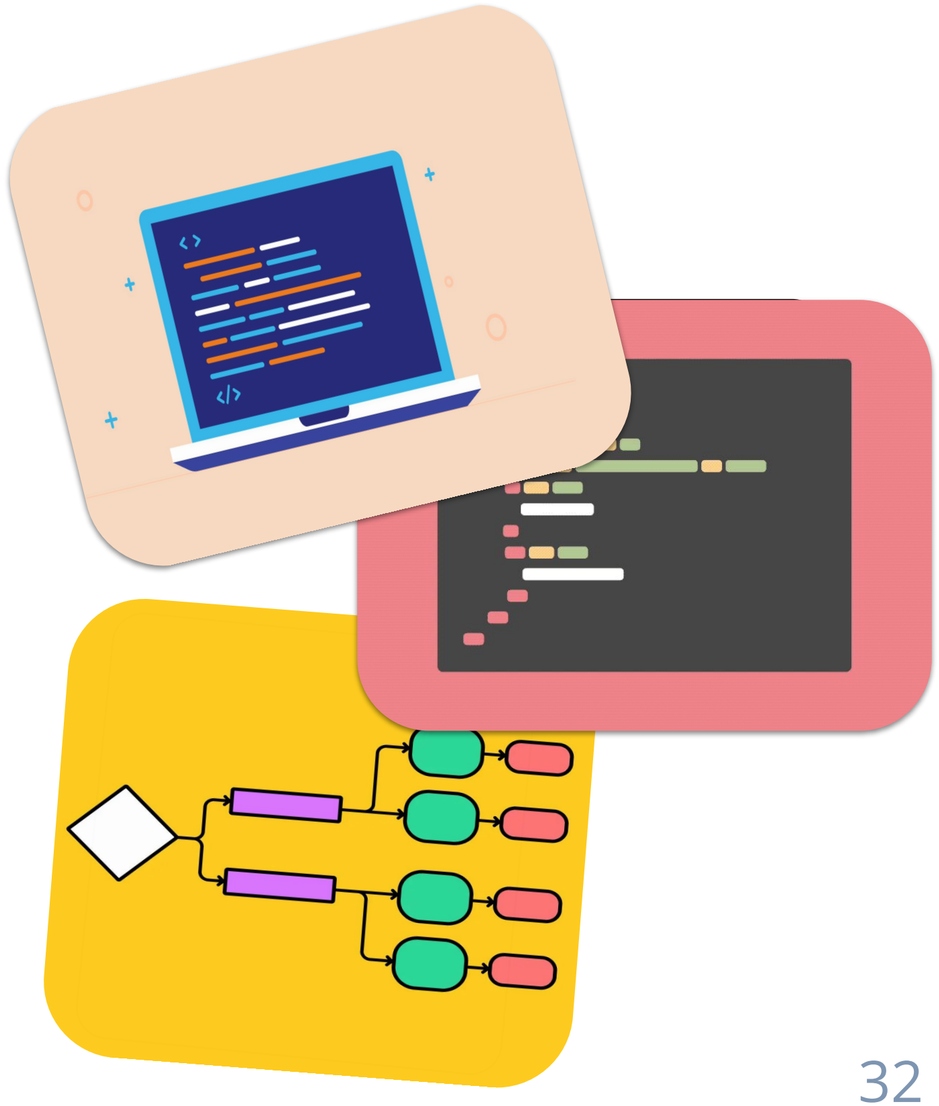
Newton School of Technology

Let’s Practice on the Playground!

Quiz Time!

Summary

- **Important List Methods**
  - **sort(), sorted(), count(), index(), reverse()**
- **List Comprehension:** A concise way to create lists using a single line with a loop and optional condition.
- **Nested Lists:** Lists within lists, often used for multi-dimensional data



Thank You!